PixelRNN Implementations: PixelCNN, Row LSTM, and Diagonal BiLSTM

Huzaifa Nasir 22I-1053

Department of Computer Science
National University of Computer and Emerging Sciences, Islamabad, Pakistan

Abstract. This report presents comprehensive implementations and evaluations of three autoregressive image modeling architectures—PixelCNN, Row LSTM, and Diagonal BiLSTM—based on the seminal PixelRNN paper. These models were trained and tested on the CIFAR-10 dataset, using discrete pixel value predictions via softmax classification. PixelCNN achieved the best validation bits-per-dimension (BPD) of 0.0018, with all models demonstrating stable convergence. Architectural trade-offs and future research directions are discussed.

1 Introduction

1.1 Problem Statement

Autoregressive image modeling approaches image generation as a sequential task, predicting each pixel conditioned on previously generated pixels. This enables explicit likelihood computation and high-quality sample synthesis, but demands architectures that balance spatial dependency modeling and computational efficiency.

1.2 Objectives

The goals of this study are:

- Implement three autoregressive architectures from the PixelRNN framework.
- Train and evaluate these models on CIFAR-10 images using discrete pixel prediction.
- Compare models using negative log-likelihood metrics, specifically bits per dimension (BPD).
- Analyze architectural trade-offs between complexity and performance.
- Validate results against theoretical benchmarks.

1.3 Dataset

The CIFAR-10 dataset includes 50,000 training and 10,000 testing RGB images of size 32×32 pixels, with discrete 8-bit pixel values (0–255). Batch size was set to 32 due to memory constraints.

2 Architectural Implementations

2.1 PixelCNN

PixelCNN employs masked convolutions with Type A and Type B masks to maintain autoregressive dependencies while allowing parallel training. The architecture includes a 7×7 kernel masked convolution followed by four residual blocks with 3×3 kernels, outputting predictions over 256 discrete pixel values. It contains approximately 44,704 parameters.

2.2 Row LSTM

Row LSTM sequentially processes images row-by-row using LSTM layers, capturing long-range intra-row dependencies. It uses a masked convolution for input projection and consists of two LSTM layers with 32 hidden channels, totaling 43,136 parameters.

2.3 Diagonal BiLSTM

Diagonal BiLSTM improves spatial modeling by processing pixels along diagonals via bidirectional LSTMs. It includes masked convolutions, one bidirectional LSTM layer with 32 hidden channels, and approximately 32,640 parameters.

3 Training Configuration

3.1 Optimization

All models were trained for 10 epochs with Adam optimizer (learning rate 1e-3), weight decay 1e-4, and ReduceLROnPlateau scheduler (factor 0.5, patience 2). Gradient clipping at max norm 1.0 ensured stability. Cross-entropy loss over pixel classes was used.

3.2 Memory Management

To manage GPU memory efficiently on Google Colab, batch sizes were limited, model hidden dimensions reduced, gradient accumulation implemented, and CUDA cache cleared regularly.

4 Results and Analysis

4.1 Training Performance

PixelCNN achieved the best validation BPD of 0.0018 after 9 epochs, followed closely by Row LSTM at 0.0019, with Diagonal BiLSTM at 0.0020. All models showed convergence with stable training dynamics, though Diagonal BiLSTM had more irregular loss patterns.

4.2 Comparative Summary

Model	Params	Final Train BPD	Final Val BPD	Best Val BPD
PixelCNN	44,704	0.0019	0.0018	**0.0018**
Row LSTM	43,136	0.0019	0.0019	0.0019
Diagonal BiLSTM	32,640	0.0020	0.0020	0.0020

4.3 Architectural Insights

PixelCNN's convolutional parallelism provides the best performance and stable training. Row LSTM benefits from sequential dependencies but at the cost of slower inference. Diagonal BiLSTM captures more complex diagonal relationships but exhibits higher training variance.

5 Challenges and Solutions

Key challenges were GPU memory constraints, numerical stability in pixel prediction, and enforcing autoregressive dependencies. These were addressed by architecture scaling, gradient clipping, masked convolutions, and efficient input processing.

6 Conclusion and Future Work

This study implemented and compared three PixelRNN architectures with PixelCNN performing best. Future research will explore larger networks, longer training schedules, self-attention mechanisms, and conditional image synthesis to further improve autoregressive modeling.